UNITED STATES DEPARTMENT OF COMMERCE
National Bureau of Standards
Washington, D.C. 20234

FIPS

1976 August 30

MEMORANDUM FOR FIPS Points of Contact
FIPSCAC

From: Harry S. White, Jr. Harry S. White, Jr.
Associate Director for ADP Standards

Subject: Review and Comment on BSR - X3.9 Draft Proposed ANS FORTRAN

My memorandum of 1976 July 30 provided for your review and comment the draft proposed American National Standard FORTRAN (BSR - X3.9).

I have just received the attached press release from ANSI concerning the review process on the draft proposal and recent changes made by the responsible technical standards committee (X3J3).

This additional material is for your information and use in preparing appropriate comments to the X3J3 Committee Secretary, Mr. Lloyd Campbell, BRL-CSD, Building 328, Aberdeen Proving Ground, Maryland 21005.

Attachment

CBEMA

COMMITTEE CORRESPONDENCE

american national standards committees:
X3—Computers & Information Processing
X4—Office Machines & Supplies
operating under the procedures of the American National Standards Institute

secretariat: CBEMA, 1828 L St NW (suite 1200), Washington DC 20036 202/466-2299

Doc. No. : X3/76-70

Date : 76-07-30
Project : 76
Milestone : 15

Reply to: X3J3 Secretary

PRESS RELEASE

"FORTRAN Standards Committee Adopts IF-THEN-ELSE"

The FORTRAN Standards Committee met in Idaho Falls, Idaho during July 12-15 to begin reviewing public comments received on the draft proposed revised FORTRAN standard. The committee, also known as X3J3, is a technical committee of the American National Standards Institute (ANSI).

At the meeting, X3J3 approved the addition of four new statements that together provide the capability to conditionally execute groups of statements. They are called block IF, ELSE IF, ELSE and END IF statements. The need for this capability was strongly presented in many of the public comments. It was also a lively topic of discussion at two public presentations on the draft standard that took place in Los Angeles in February and Washington, D.C. in March.

X3J3 published its draft proposal in the March issue of SIGPLAN Notices, a publication of the Special Interest Group on Programming Languages of the Association for Computing Machinery. More than eight thousand copies have been distributed to interested individuals, and technical, business and governmental organizations around the world.

The widespread interest in the proposal for a revised FORTRAN standard is indicated by the substantial volume of comments received. As of the beginning of the meeting, 200 letters had been received totaling 810 pages. The overwhelming majority of comments are favorable and contain many constructive suggestions. According to ANSI procedures, each suggestion will be evaluated to determine whether a change should be made to the draft standard. Following completion of the X3J3 review process, each public review letter will be answered indicating the action taken.

X3J3 will continue its review of public comments at its next meeting in September. The public review and comment period closes September 28, 1976.

(Approved for release by
X3J3 on 15 July 1976)

Bell Laboratories

subject FORTRAN Standards Committee Adopts
IF-THEN-ELSE

date July 22, 1976

from: J. C. Noll

To X3J3:

IF-THEN-ELSE was adopted at the July meeting of X3J3. The attached press release and IF-THEN-ELSE text is being sent to you as information relating to the processing of dpANS FORTRAN.

Press Release

Attached is a press release announcing the adoption of IF-THEN-ELSE by X3J3. IF-THEN-ELSE was adopted for the FORTRAN full language and the subset language.

IF-THEN-ELSE Text

The principal change to the dpANS FORTRAN is to Section 11, CONTROL STATEMENTS. Section 11 of Document X3J3/76.3 FORTRAN Full Language is attached. The text of the subset is not attached since the IF-THEN-ELSE subset text is identical to that of the full language.

Document X3J3/76 remains the basis document for dpANS FORTRAN. Document X3J3/76.3 is a working document of X3J3 and is subject to further changes.

Comments on dpANS FORTRAN or the new IF-THEN-ELSE text may be sent to:

Lloyd W. Campbell
X3J3 Secretary
BRL-CSD Bldg. 328
Aberdeen Proving Ground
MD 21005 USA

HO-8223-JCN-dg

J. C. Noll

Att.
Press Release
X3J3/76.3 Section 11,
CONTROL STATEMENTS

11. CONTROL STATEMENTS

Control statements may be used to control the execution sequence.

There are sixteen control statements:

(1) Unconditional GO TO

(2) Computed GO TO

(3) Assigned GO TO

(4) Arithmetic IF

(5) Logical IF

(6) Block IF

(7) ELSE IF

(8) ELSE

(9) END IF

(10) DO

(11) CONTINUE

(12) STOP

(13) PAUSE

(14) END

(15) CALL

(16) RETURN

CALL and RETURN statements are described in Section 15.

11.1 Unconditional GO TO Statement

The form of an unconditional GO TO statement is:

```
GO TO s
```

where s is the statement label of an executable statement that appears in the same program unit as the unconditional GO TO statement.

Execution of an unconditional GO TO statement causes a transfer of control so that the statement identified by the statement label is executed next.

11.2 Computed GO TO Statement

The form of a computed GO TO statement is:

```
GO TO (s [,s]... ) [,] e
```

where: e is an integer, real, or double precision expression

s is the statement label of an executable statement that appears in the same program unit as the computed GO TO statement. The same statement label may appear more than once in the same computed GO TO statement.

Execution of a computed GO TO statement causes evaluation of the expression INT(e). Let the value of INT(e) ba i. The eveluation of INT(e) is followed by a transfer of control so that the statement identified by the ith statement label in the list of statement labels is executed next, provided that 1 ≤ i ≤ n, where n is the number of statement labels in the list of statement labels. If i<1 or i>n, the execution sequence continues as though a CONTINUE statement were executad.

11.3 Assigned GO TO Statement

The form of an assigned GO TO statement is:

```
GO TO i [[,] ( s [,s]... )]
```

where: i is an integer variable name

s is the statement label of an executable statement that appears in the same program unit as the assigned GO TO statement. The same statement label may appear more than once in the same assigned GO TO statement.

At the time of execution of en assigned GO TO stetement, the current value of i must have been essigned by the prior execution of an ASSIGN statement (10.3) to the statement lebel of en executable statement. The execution of the assigned GO TO statement causes a transfer of control so that the statament identified by that statement label is executed next. The last definition of the variable in en assigned GO TO statement must have occurred in the same program unit as the assigned GO TO statament.

If the parenthesized list is present, the statement label assigned to i must be one of the statement labels in the list.

11.4 Arithmetic IF Statement

The form of an arithmetic IF statamant is:

```
IF ( e ) s1 , s2 , s3
```

where: e Is an integer, real, or double precision expression

s_1, s_2, and s_3 are each the statement label of an executable statement thet appears in the same program unit as the arithmetic IF statement. The same statemant label may appeer more than once in the seme arithmetic IF statement.

Execution of an arithmetic IF statement causes evaluation of the expression e followed by a transfer of control. The statement identified by s_1, s_2, or s_3 is executed next as tha value of e is less than zero, equal to zero, or greater than zero, respectively.

11.5 Logical IF Statement

The form of a logical IF statement is:

```
IF ( e ) st
```

where: e is a logical expression

st Is eny executeble statement except a DO, block IF, ELSE IF, ELSE, ENO IF, END, or anothar logical IF statemant.

Execution of a logical IF statement ceuses eveluetion of the axpression e. If the velue of e is true, statement st is executed. If the valua of e is felse, statement st is not executed end the execution sequence continues as though a CONTINUE statement were executed.

Note that the execution of a function referenca in the expression e of a logical IF statement is permitted to effect entities in the statemant st.

11.6 Block IF Statement

The block IF statement is used with the ENO IF stetement and, optionelly, tha ELSE IF end ELSE statements to control the execution sequence.

The form of a block IF statement is:

```
IF (e) THEN
```

where e is a logical expression.

11.6.1 IF-level

The IF-level of a statement s is

$$n_1 - n_2$$

where n_1 is the number of block IF statements from the beginning of the program unit up to and including s, and n_2 is the number of END IF statements in the program unit up to but not including s.

The IF-level of every statement must be zero or positive. The IF-level of each block IF, ELSE IF, ELSE, and END IF statement must be positive. The IF-level of the END statement of each program unit must be zero.

11.6.2 IF-Block

An IF-block consists of all of the executable statements after the block IF statement up to, but not including, the next ELSE IF, ELSE, or END IF statement that has the same IF-level as the block IF statement. An IF-block may be empty.

11.6.3 Execution of a Block IF Statement

Execution of a block IF statement causes evaluation of the expression e. If the value of e is true, normal execution sequence continues with the first statement of the IF-block. If the IF-block is empty, control is transferred to the next END IF statement that has the same IF-level as the block IF statement. If the value of e is false, control is transferred to the next ELSE IF, ELSE, or END IF statement that has the same IF-level as the block IF statement.

Transfer into an IF-block is permitted.

If the execution of the last statement in the IF-block does not result in a transfer of control, control is transferred to the next END IF statement that has the same IF-level as the block IF statement that precedes the IF-block.

11.7 ELSE IF Statement

The form of an ELSE IF statement is:

ELSE IF (e) THEN

where e is a logical expression.

11.7.1 ELSE IF-Block

An ELSE IF-block consists of all of the executable statements after the ELSE IF statement up to, but not including, the next ELSE IF, ELSE, or END IF statement that has the same IF-level as the ELSE IF statement. An ELSE IF-block may be empty.

11.7.2 Execution of an ELSE IF Statement

Execution of an ELSE IF statement causes evaluation of the expression e. If the value of e is true, normal execution sequence continues with the first statement of the ELSE IF-block. If the ELSE IF-block is empty, control is transferred to the next END IF statement that has the same IF-level as the ELSE IF statement. If the value of e is false, control is transferred to the next ELSE IF, ELSE, or END IF statement that has the same IF-level as the ELSE IF statement.

Transfer into an ELSE IF-block is permitted.

If execution of the last statement in the ELSE IF-block does not result in a transfer of control, control is transferred to the next END IF statement that has the same IF-level as the ELSE IF statement that precedes the ELSE IF-block.

11.8 ELSE Statement

The form of an ELSE statement is:

ELSE

11.8.1 ELSE-Block

An ELSE-block consists of all of the executable statements after the ELSE statement up to, but not including, the next END IF statement that has the same IF-level as the ELSE statement. An ELSE-block may be empty.

An END IF statement of the same IF-level as the ELSE statement must appear before the appearance of an ELSE IF or ELSE statement of the same IF-level.

11.8.2 Execution of an ELSE Statement

Execution of an ELSE statement has no effect. Normal execution sequence continues.

Transfer into an ELSE-block is permitted.

11.9 END IF Statement

The form of an END IF statement is:

END IF

Execution of an END IF statement has no effect. Normal execution sequence continues.

For each block IF statement, there must be e corresponding END IF statement in the same program unit. A corresponding END IF statement is the next ENO IF statement that hes the same IF-level as the block IF statement.

11.10 DO Statement

A 00 statement is used to specify a loop, celled a DO-loop.

The form of a 00 statement is:

```
          00 s [,] i = e₁, e₂ [,e₃]
```

where: s is the statement label of an executable statement. The statement identified by s, called the terminal statement of the 00-loop, must physically follow and appear in the same program unit as the 00 statement.

i is the name of an integer, real, or double precision variable, called the DO-variable

e_1, e_2, and e_3 are each an integer, real, or double precision expression.

The terminal statement of e DO-loop must not be en unconditional GO TO, assigned GO TO, erithmetic IF, block IF, ELSE IF, ELSE, END IF, RETURN, STOP, ENO, or DO statement. If the terminal statement of a 00-loop is a logical IF, it mey contain any executable statement except a 00, block IF, ELSE IF, ELSE, END IF, END, or another logical IF statement.

11.10.1 Range of a DO-Loop

The range of a DO-loop consists of the executable statements from and including the first executable stetement following the DO statement that specifies the DO-loop, to and including the terminal statement of the 00-loop.

If a DO statement appears within the range of a 00-loop, the range of the DO-loop specified by that DO statement must be within the range of the outer DO-loop. More than one DO-loop may have the same terminal statement.

If a DO statement appears within an IF-block, ELSE IF-block, or ELSE-block, the range of that DO-loop must be conteined entirely within that IF-block, ELSE IF-block, or ELSE-block, respectively.

If a block IF statement appears within the range of a 00-loop, the corresponding END IF statement must also appear within the range of that DO-loop.

11.10.2 Active and Inactive DO-Loops

A 00-loop is aither active or inactive. Initielly inactive, a 00-loop becomes active only when its 00 statement is executed.

Once ective, the 00-loop becomes inactive only when:

(1) its iteration count is zero,

(2) its DO-variable becomes undefined or is redefined by means other than the incrementation described in 11.10.7,

(3) a RETURN, STOP, or END stetement is executed in its program unit,

(4) it is in tha range of another 00-loop that becomes inactive, or

(5) it is in the range of another 00-loop whose 00 stetement is executed.

Note that transfer of control out of the range of a DO-loop does not inactivate the DO-loop. However, the DO-loop becomes inactive if the 00-verieble becomes undefined or is redefined outside the range.

When a DO-loop becomes inactive, the 00-variable of the DO-loop retains its last defined value unless it has become undefined.

11.10.3 Executing a DO Statement

Tha affect of executing a DO statement is to perform tha following staps in sequence:

(1) The initial parameter m_1, the terminal parameter m_2, and the incrementation parameter m_3 are established by evaluating e_1, e_2, and e_3, respectively, including, if necessary, conversion to the type of the DO-variable according to the rules for arithmetic conversion (Table 4). If e_3 does not appear, m_3 has a value of one. m_3 must not have e velue of zero.

(2) The DO-varieble becomes defined with the value of the initial parameter m_1.

(3) The iteration count is established and is the value of the expression

$$\text{MAX(INT(}(m_2 - m_1 + m_3)/m_3\text{), 0)}$$

Note that the iteration count is zero whenever:

$\underline{m}_1 > \underline{m}_2$ and $\underline{m}_3 > 0$, or

$\underline{m}_1 < \underline{m}_2$ and $\underline{m}_3 < 0$.

At the completion of execution of the DO statement, loop control processing begins.

11.10.4 Loop Control Processing

Loop control processing determines if further execution of the range of the DO-loop is required. The iteration count is tested. If it is not zero, execution of the first statement in the range of the DO-loop begins. If the iteration count is zero, the DO-loop becomes inactive. If, as a result, all of the DO-loops sharing the terminal statement of this DO-loop are inactive, normal execution continues with execution of the next executable statement following the terminal statement. However, if some of the DO-loops sharing the terminal statement are active, execution continues with incrementation processing, as described below.

11.10.5 Execution of the Range

Statements in the range of a DO-loop are executed until the terminal statement is reached. Except by the incrementation described in 11.10.7, the DO-variable of the DO-loop may neither be redefined nor become undefined during execution of the range of the DO-loop.

11.10.6 Terminal Statement Execution

Execution of the terminal statement occurs as a result of the normal execution sequence or as a result of transfer of control, subject to the restrictions in 11.10.8. Unless execution of the terminal statement results in a transfer of control, execution then continues with incrementation processing, as described below.

11.10.7 Incrementation Processing

Incrementation processing has the effect of the following steps performed in sequence:

(1) The DO-variable, and the incrementation parameter of the active DO-loop whose DO statement was most recently executed, are selected for processing.

(2) The value of the DO-variable is incremented by the value of the incrementation parameter $\underline{m}_3$.

(3) The iteration count is decremented by one.

(4) Execution continues with loop control processing (11.10.4) of the same DO-loop whose iteration count was decremented.

An example illustrates the above:

```
          N=0
          DO 100 I=1,10
          J=I
          DO 100 K=1,5
          L=K
      100 N=N+1
      101 CONTINUE
```

After execution of the above statements and at the execution of the CONTINUE statement, I=11, J=10, K=6, L=5, and N=50.

Also consider the following example:

```
          N=0
          DO 200 I=1,10
          J=I
          DO 200 K=5,1
          L=K
      200 N=N+1
      201 CONTINUE
```

After execution of the above statements and at the execution of the CONTINUE statement, I=11, J=10, K=5, and N=0. L is not defined by the above statements.

11.10.8 Transfer into the Range of a DO-Loop

Transfer of control into the range of an inactive DO-loop is not permitted. Transfer of control to any executable statement in the range of an active DO-loop is permitted unless the statement is also in the range of an inactive DO-loop.

11.11 CONTINUE Statement

The form of a CONTINUE statement is:

```
CONTINUE
```

Execution of a CONTINUE statement has no effect.

If the CONTINUE statement is not the terminal statement of a DO-loop, normal execution sequence continues. If the CONTINUE statement is the terminal statement of a DO-loop, the next statement executed depends on the result of the DO-loop incrementation processing (11.10.7).

11.12 STOP Statement

The form of a STOP statement is:

STOP [$\underline{n}$]

where _n_ is a string of not more than five digits, or is a character constant.

Execution of a STOP statement causes termination of execution of the executable program. At the time of termination, the digit string or character constant is accessible.

11.13 PAUSE Statement

The form of a PAUSE statement is:

```
          PAUSE [n]
```

where _n_ is a string of not more than five digits, or is a character constant.

Execution of a PAUSE statement causes a cessation of execution of the executable program. Execution must be resumable. At the time of cessation of execution, the digit string or character constant is accessible. Resumption of execution is not under control of the program. If execution is resumed, the normal execution sequence is continued.

11.14 END Statement

The END statement indicates the end of the sequence of statements and comment lines of a program unit (3.5). If executed in a subprogram, it has the effect of a RETURN statement (15.8). If executed in a main program, it terminates the execution of the executable program.

The form of an END statement is:

```
          END
```

An END statement is written only in columns 7 through 72 of an initial line. An END statement must not be continued. No other statement in a program unit may have an initial line that appears to be an END statement.

The last line of every program unit must be an END statement.